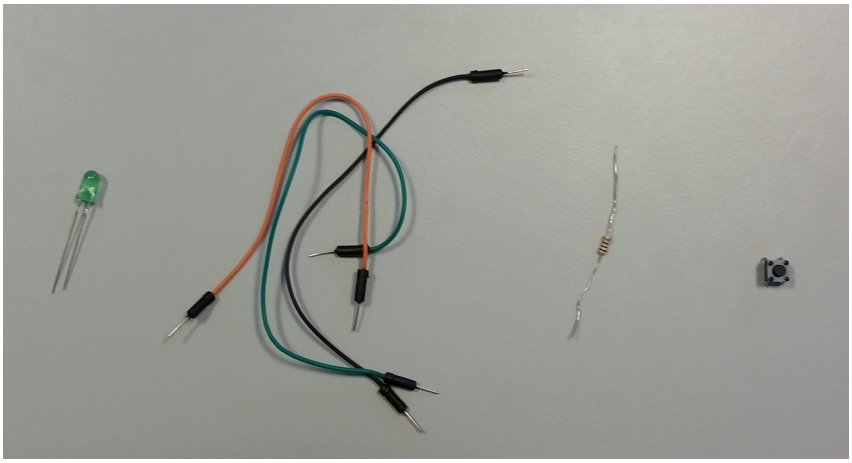
Programming the Arduino

The small blue board in front of you is called an Arduino board. Arduino is a device called a *microcontroller.* This allows it to control any of the devices that you will connect to it (lights, switches, wires etc) and make them do anything you want. These exercises will teach you how to make it do that, by programming the board. They will start off with very easy control, making an LED blink, and eventually will build up to a game you can play which can demonstrate DNA splicing. Ask one of our team in case you need any help, or you have any questions about this exercise.

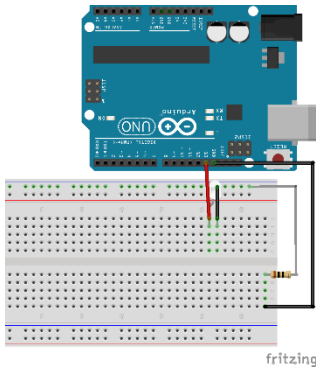
The Arduino should already be plugged into the computer for you and the computer should already be setup with the Arduino specific text editor open.

Throughout these exercises you will be using:

LEDs Wires Resistor Button



Exercise 1 – blink



This exercise involves:

1x LED 4x wires 1x resistor

This is the first exercise that you can do with Arduino. The circuit components this will use are shown on the right – you will need to connect the wires and set the board up yourselves.

Now **go to File -> Examples -> 01.Basics -> Blink** and the file blink code will load. This is shown below.

All the text in grey are comments and doesn't actually do anything, so it can be safely ignored.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://arduino.cc

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

You'll notice the program is split into two main parts, called `setup` and `loop`. `setup` runs once at the beginning of the program, while `loop` runs repeatedly after that. All that `setup` does is enable pin 13 on the Arduino to be an output pin – this will let the pin turn the LED on and off.

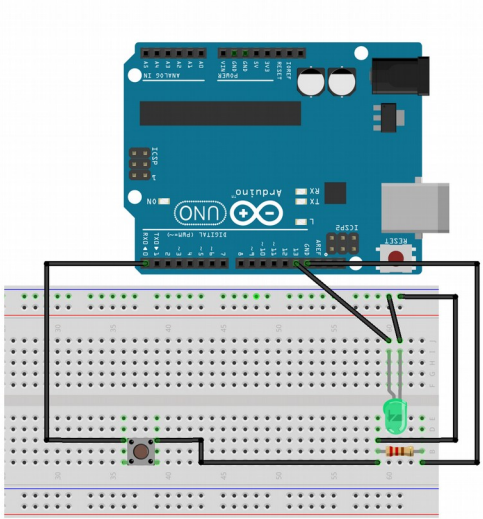
`loop` runs repeatedly, and it is this part of the program that makes the LED blink. The `digitalWrite` command allows the pin to be set to a high voltage (`HIGH`, which turns on the LED) or a low voltage (`LOW`, which turns it off). The `delay` command makes the Arduino do nothing for the number of milliseconds specified – here, 1000 milliseconds, which is 1 second.

Once you're satisfied you understand how the program works, **go to File -> Upload** or press the Upload button near the top-left. After a few seconds an LED on the board should be flashing.

Exercise 2 – button

This exercise involves:

1x LED 6x wires 2x resistors 1x button



This exercise should introduce you to conditional logic, which is essential in all programming. As well as the LED, this exercise will use the push button as well.

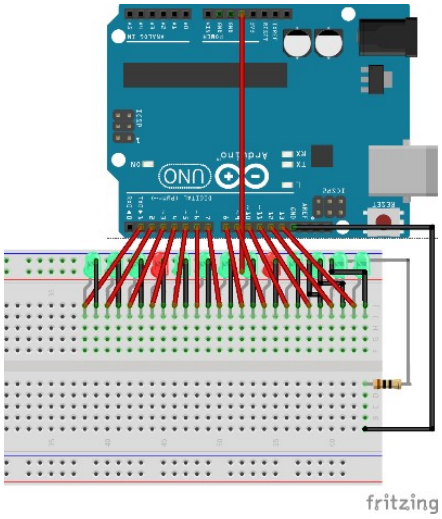
Go to File -> Examples -> 02.Digital -> Button and open the file.

This program will turn the button into a light switch, turning the LED on when you push the button – even though there is no physical connection between the button and the LED.

Look at the code in `setup` now. The LED pin is set up as before, but now the button pin is set to be `INPUT`. This changes its function, making it read whether the voltage on the pin is high enough.

The code in loop now repeatedly checks whether the button has been pushed using the `digitalRead` command. Then it switches the LED on and off using an `if - else` statement. This is a very simple example of *conditional logic*: allowing the computer to behave differently in different circumstances, which is what makes computers so powerful.

Now upload the code as you did before and experiment a bit with the button to make sure you understand how it works.



Exercise 3 – LED row

This exercise involves:

13x LEDs 31x wires 1x resistor

This program will introduce you to the concept of loops, which is another very important and powerful tool in computing. This program will use the full row of LEDs on your board.

The Arduino for this exercise and the next exercise has been set up for you. It has 13 LEDs. Unplug the current Arduino and plug in the second Arduino.

Go to My Documents and open the file “Row_-_for_loop.ino”. This should show you the code displayed below.

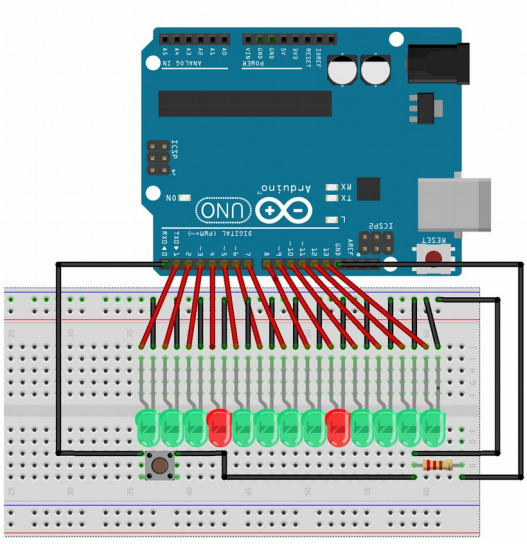
```
/*
  Blink Entire Row
  Turns on an LED on for 200 milliseconds, then turns on next LED.
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize all digital pins as output pins. - pins 1 to 13
  for (int i = 1; i<14; i = i + 1){
    pinMode(i, OUTPUT);
  }

}

// the loop function runs over and over again forever
void loop() {
  //turn on each led in turn
  for(int i = 1; i<14; i = i + 1){
    digitalWrite(i, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(200);              // wait for a 200 milliseconds
    digitalWrite(i, LOW);    // turn the LED off by making the voltage LOW
  }
}
```

Take a look at the code in `setup`. This sets all the pins to `OUTPUT`, but does this in a different way to before. Rather than take up 13 lines to set each pin individually, it uses what is known as a `for` loop. It defines initially `i = 1`, and then sets pin `i` to OUTPUT. This sets the first pin (pin `i` - currently pin1) as an OUTPUT pin. Once that is done, it increases `i` by one and repeats until the value of `i` is no longer less than 14 (i.e. until `i` is 13), which sets all the pins.

The code in `loop` also uses a `for` loop. This loop makes it cycle through all the LEDs and set each one up individually. Study the code and make sure you are satisfied that the way it does this is exactly the same as the way it did it in setup.

Upload the code as before. The next exercise involves you coding your own if statements and for loops, so make sure you understand them fully. If you have any questions, ask a team member – that's why we're here!

Exercise 4 – Splicing Game



This exercise involves:

13x LEDs 31x wires 1x resistor 1x button

This exercise combines everything you have learned so far. We will add a button to the setup of exercise 3 and then add some 'logic' to make a game!

Go to My Documents and open the file "Outreach_Game_toFinish.ino". For this exercise we have already written most of the code, however we have left out some of the code in the for loops for you to complete. You will need to replace any sections that have several question marks before and after the text, such as:

?????Turn the current LED on?????

In order to complete the game, you'll need to know how the game should work! So, the game is a gene-cutter game. If you want to take some trait from one organism to another, you would want to take the gene that is responsible for this trait and put it into the DNA of the second organism. Our game is simple – the idea is you have a light moving back and forth along a row of LEDs (a section of DNA). Two of these LEDs are red and these represent the end of the gene. You need to click the button at the points you want to cut the DNA (at the two red LEDs to cut the gene properly). Each time you succeed, the game speeds up.